Introduction to Provable Security in Public-Key Cryptography

Damien Vergnaud
(Mathematical Foundations of Asymmetric Cryptography)

Sorbonne Université – CNRS – IUF

Contents

1 Introduction

2 Public-Key Encryption
   • Definitions
   • Security Notions for Public-Key Encryption

3 Discrete-log based encryption schemes
   • ElGamal encryption scheme
   • Random Oracle Model and Variants of ElGamal

4 Digital signatures
   • Definitions
   • Security Notions for Digital Signatures

5 Discrete-log based digital signatures
   • One-time signatures
   • Fiat-Shamir heuristic and Schnorr Signatures
Cryptography

**Goal:** enable “secure” communication in the presence of adversaries

internet, phone line, ...

Alice

Bob

eavesdrops

Eve

Encryption

Alice sends a ciphertext to Bob
Only Bob can recover the plaintext

CONFIDENTIALITY

To recover the plaintext

- to find the whole plaintext ?
- to get some information about it ?

Which means can be used ?

- just the ciphertext ?
- some extra information ?
Why “Provable Security” ?

Once a cryptosystem is described, how can we prove its security?

- by trying to exhibit an attack
  - attack found ⇒ system insecure!
  - attack not found ⇒ ?

- by proving that no attack exists under some assumptions
  - attack found ⇒ false assumption

- "Textbook" cryptosystems cannot be used as such

- Practitioners need formatting rules to ensure operability.
  --> Paddings are used in practice: heuristic security

- Provable security is needed in upcoming systems.
  This is no longer just theory.

- Provable security is fun! :-)

Who is the bad guy?

We are protecting ourselves from the evil Eve, who

- is a probabilistic polynomial time Turing machine (PPTM) (Church-Turing thesis)

- knows all the algorithms (Kerckoff’s principles)

- has full access to communication media.
Proof by reduction

A adversary against e.g. one-wayness

Proof by reduction

A adversary against e.g. one-wayness
Proof by reduction

Instance $I$ of a problem $\mathcal{P}$

\[ A \]

Proof by reduction

Instance $I$ of a problem $\mathcal{P}$

\[ A \]

Solution of $I$
Proof by reduction

Instance $I$ of a problem $\mathcal{P}$

$R$

Solution of $I$

$\mathcal{P}$ intractable $\rightarrow$ contradiction

The Methodology of “Provable Security”

- Define goal of adversary
- Define security model
- Define complexity assumptions
- Provide a proof by reduction
- Check proof
- Interpret proof
Secret-Key Encryption

Symmetric encryption: Alice and Bob share a “key” $K$

- Bob can use the same method to send messages to Alice.
  - \(\rightarrow\) symmetric setting
- How did Alice and Bob establish $K$?

The solution: Public-Key Cryptography

- first proposed by Diffie and Hellman:

  W. Diffie and M. E. Hellman,  
  New directions in cryptography  

- 2015 Turing Award
- It 1997 the GCHQ revealed that they knew it already in 1970 (James Ellis).
Public-Key Encryption

**Asymmetric encryption:** Bob owns two “keys”
- a public key
- a secret key

Alice

\[ \text{pk}_B \rightarrow \text{sk}_B \]

Bob

\[ ?? \]

But is it possible?

- In “physical world”: yes!
  - Example: padlock

\[ \text{anyone can lock it} \rightarrow \text{the key is needed to unlock} \]

Diffie and Hellman proposed the public-key cryptography in 1976.
- They just proposed the concept, not the implementation.
- But they have shown a protocol for key-exchange
Diffie-Hellman Key Exchange

$(\mathbb{G}, \cdot)$ a finite cyclic group; $\langle g \rangle = \mathbb{G}$

Eve knows:

- $(\mathbb{G}, g)$
- $y_a = g^a$
- $y_b = g^b$

and should have “no information” on $K = g^{ab}$.

- If finding $a$ from $y_a$ is easy then the DH key exchange is not secure.
- Even if it is hard, then
  
  \[ K_a = y_b^a = (g^b)^a = g^{ab} = (g^a)^b = y_a^b = K_b \]

- the scheme may also not be completely secure

- How to choose the group $\mathbb{G}$?
  
  see Pierrick’s lectures

- Do we really need a group?
  
  see Luca’s lectures
Public-Key Encryption

An asymmetric encryption scheme is a triple of algorithms \((\mathcal{K}, \mathcal{E}, \mathcal{D})\) where

- \(\mathcal{K}\) is a probabilistic key generation algorithm which returns random pairs of secret and public keys \((sk, pk)\) depending on the security parameter \(\kappa\),

- \(\mathcal{E}\) is a probabilistic encryption algorithm which takes on input a public key \(pk\) and a plaintext \(m \in \mathcal{M}\), runs on a random tape \(u \in \mathcal{U}\) and returns a ciphertext \(c\),

- \(\mathcal{D}\) is a deterministic decryption algorithm which takes on input a secret key \(sk\), a ciphertext \(c\) and returns the corresponding plaintext \(m\) or the symbol \(\bot\).

If \((sk, pk) \leftarrow \mathcal{K}\), then \(\mathcal{D}_{sk}\) \((\mathcal{E}_{pk}(m, u))\) = \(m\) for all \((m, u) \in \mathcal{M} \times \mathcal{U}\).

Encryption: Security Notions

Encryption is supposed to provide confidentiality of the data.

But what exactly does this mean?

<table>
<thead>
<tr>
<th>Security goal</th>
<th>But . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recovery of secret key is infeasible</td>
<td>True if data is sent in the clear</td>
</tr>
<tr>
<td>Obtaining plaintext from ciphertext is infeasible</td>
<td>Might be able to obtain half the plaintext etc</td>
</tr>
</tbody>
</table>

So what is a secure encryption scheme?

Not an easy question to answer . . .

Attackers should not be able to compute any information about \(m\).
How to formalize it?

Attackers should not be able to compute any information about \( m \).

Probabilistic approach
- \( M \) some random variable that takes values from \( \mathcal{M} \)
- \( K \) random variable distributed uniformly over \( \mathcal{K} \)
- \( C = \mathcal{E}_K(M) \)

Definition
An encryption scheme is perfectly secret if for every random variable \( M \) and every \( m \in \mathcal{M} \) and every \( c \in \mathcal{C} \) with \( \Pr(C = c) > 0 \):

\[
\Pr(M = m) = \Pr(M = m | C = c)
\]

\( \bowtie \) \( C \) and \( M \) are independent

A perfectly symmetric secure scheme: one-time pad

Description
- \( \ell \in \mathbb{N} \) a parameter. \( \mathcal{M} = \mathcal{K} = \{0, 1\}^\ell \).
- Let \( \oplus \) denote component-wise XOR.
- **Vernam’s cipher**: \( \text{Enc}(K, m) = m \oplus K \) and \( \text{Dec}(K, c) = c \oplus K \).

- One-time pad is **perfectly secret**!

\[
\Pr(C = c | M = m) = \Pr(K \oplus M = c | M = m) = \Pr(K = m \oplus c | M = m) = 2^{-\ell}
\]

- Each key cannot be used **more than once**!

\[
\text{Enc}(K, m_0) \oplus \text{Enc}(K, m_1) = (m_0 \oplus K) \oplus (m_1 \oplus K) = m_0 \oplus m_1
\]

- One time-pad is **optimal** in the class of perfectly secret schemes
Security Notions

Depending on the context in which a given cryptosystem is used, one may formally define a security notion for this system,

- by telling what **goal** an adversary would attempt to reach,
- and what means or information are made available to her (the **model**).

A security notion (or level) is entirely defined by pairing an adversarial goal with an adversarial model.

**Examples:** OW-PCA, IND-CCA2, NM-CCA2.

History of Security Goals

- it shouldn’t be feasible to compute the secret key $sk$ from the public key $pk$ (**unbreakability** or UBK). Implicitly appeared with public-key crypto.

- it shouldn’t be feasible to invert the encryption function over any ciphertext under any given key $pk$ (**one-wayness** or OW). Diffie and Hellman, late 70’s.

- it shouldn’t be feasible to recover a **single bit of information** about a plaintext given its encryption under any given key $pk$ (**semantic security** or SEM). Goldwasser and Micali, 1982.

- it shouldn’t be feasible to distinguish pairs of ciphertexts based on the message they encrypt (**indistinguishability** or IND). Goldwasser and Micali, 1982.

- it shouldn’t be feasible **to transform** some ciphertext into another ciphertext such that plaintext are meaningfully related (**non-malleability** or NM). Dolev, Dwork and Naor, 1991.
History of Adversarial Models

Several types of computational resources an adversary has access to have been considered:

- **chosen-plaintext attacks** (CPA), unavoidable scenario.

- **non-adaptive chosen-ciphertext attacks** (CCA1), wherein the adversary gets, in addition, access to a decryption oracle before being given the challenge ciphertext. Naor and Yung, 1990.

- **adaptive chosen-ciphertext attacks** (CCA2) as a scenario in which the adversary queries the decryption oracle before and after being challenged; her only restriction here is that she may not feed the oracle with the challenge ciphertext itself. Rackoff and Simon, 1991.

Semantic Security

Semantic security for $\mathcal{E} = (G, E, D)$, against an adversary $\mathcal{A}$ and attack $atk \in \{\text{cpa, cca1, cca2}\}$ is measured using the following game:

Experiment $\text{Expt}^{\text{sem-atk-b}}_{\mathcal{E}}(\mathcal{A}, \kappa)$:

1. $(pk, skK) \leftarrow G(1^n)$;
2. $(\mathcal{M}, s) \leftarrow \mathcal{A}^{D_0(\cdot)}(\text{select}, pk)$;
3. $x_0 \overset{R}{\leftarrow} \mathcal{M}$; $x_1 \overset{R}{\leftarrow} \mathcal{M}$;
4. $y \leftarrow E_{pk}(x_b)$;
5. $(f, \alpha) \leftarrow \mathcal{A}^{D_1(\cdot)}(\text{predict}, y, s)$;
6. if $f(x_b) = \alpha$ then return 1;
7. else return 0;

- $\mathcal{M}: \mathcal{P} \rightarrow [0, 1]$ is a **distribution** over the plaintext space
- $f: \mathcal{P} \rightarrow \text{ran } f$ is a **function** on plaintexts, with $\alpha \in \text{ran } f$.
- The **oracles** $D_0$ and $D_1$ are defined according to $atk$:

<table>
<thead>
<tr>
<th>atk</th>
<th>$D_0(x)$</th>
<th>$D_1(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPA</td>
<td>$\bot$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>CCA1</td>
<td>$D_{sk}(x)$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>CCA2</td>
<td>$D_{sk}(x)$</td>
<td>$D_{sk}(x)$ for $x \neq y$</td>
</tr>
</tbody>
</table>
Indistinguishability

Indistinguishability for $\mathcal{E} = (G, E, D)$, against an adversary $A$ and attack $atk \in \{\text{cpa, cca1, cca2}\}$ is measured using the following game:

$$\text{Experiment } \text{Expt}^{\text{ind-atk-b}}_{\mathcal{E}}(A, \kappa):$$

1. $(pk, skK) \leftarrow G(1^\kappa)$;
2. $(x_0, x_1, s) \leftarrow A^{D_0(\cdot)}(\text{find, pk})$;
3. if $|x_0| \neq |x_1|$ then return 0;
4. $y \leftarrow E_{pk}(x_b)$;
5. $b' \leftarrow A^{D_1(\cdot)}(\text{guess, y, s})$;
6. return $b'$;

- In the first stage, the adversary has to choose two plaintexts.
- One is encrypted by the challenger and the ciphertext given to the adversary.
- The adversary must decide which plaintext was encrypted.
- The oracles $D_0$ and $D_1$ are defined according to $atk$:

<table>
<thead>
<tr>
<th>$atk$</th>
<th>$D_0(x)$</th>
<th>$D_1(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPA</td>
<td>⊥</td>
<td>⊥</td>
</tr>
<tr>
<td>CCA1</td>
<td>$D_{sk}(x)$</td>
<td>⊥</td>
</tr>
<tr>
<td>CCA2</td>
<td>$D_{sk}(x)$</td>
<td>$D_{sk}(x)$ for $x \neq y$</td>
</tr>
</tbody>
</table>

Advantage and insecurity

For a public-key encryption scheme $\mathcal{E}$, under attack $atk \in \{\text{cpa, cca1, cca2}\}$ by an adversary $A$, we define $A$’s advantage by:

$$\text{Adv}^{\text{ind-atk}}_{\mathcal{E}}(A) = \Pr[\text{Expt}^{\text{ind-atk-1}}_{\mathcal{E}}(A) = 1] - \Pr[\text{Expt}^{\text{ind-atk-0}}_{\mathcal{E}}(A) = 1];$$
$$\text{Adv}^{\text{sem-atk}}_{\mathcal{E}}(A) = \Pr[\text{Expt}^{\text{sem-atk-1}}_{\mathcal{E}}(A) = 1] - \Pr[\text{Expt}^{\text{sem-atk-0}}_{\mathcal{E}}(A) = 1].$$

We define insecurities for $\text{goal} \in \{\text{ind, sem}\}$ under chosen plaintext attacks, and chosen ciphertext attacks $cca \in \{\text{cca1, cca2}\}$ by:

$$\text{InSec}^{\text{goal-cpa}}_{\mathcal{E}}(t) = \max_A \text{Adv}^{\text{goal-cpa}}_{\mathcal{E}}(A);$$
$$\text{InSec}^{\text{goal-cca}}_{\mathcal{E}}(t, q_D) = \max_A \text{Adv}^{\text{goal-cca}}_{\mathcal{E}}(A).$$

where the maxima are taken over adversaries $A$ which run in time $t$ and issue $q_D$ decryption queries.
Chosen-Ciphertext Security

Because IND-CCA2 ≡ NM-CCA2 is the upper security level, it is desirable to prove security with respect to this notion. It is also denoted by IND-CCA and called chosen ciphertext security.

- Formally, an asymmetric encryption scheme is said to be \((\tau, \varepsilon)\)-IND-CCA if for any adversary \(A = (A_1, A_2)\) with running time upper-bounded by \(\tau\),

\[
\text{Adv}^{\text{ind}}_E(A) = 2 \times \Pr_{b \in \{0, 1\}, u \in \{0, 1\}^k} \left[ \begin{array}{l}
(sk, pk) \leftarrow \mathcal{K}(1^n), (m_0, m_1, \sigma) \leftarrow A_1(pk),
\ c \leftarrow E_{pk}(m_b, u), A_2(c, \sigma) = b
\end{array} \right] - 1 < \varepsilon,
\]

where the probability is taken over the random choices of \(A\).

- The two plaintexts \(m_0\) and \(m_1\) chosen by the adversary have to be of identical length.

- Access to a decryption oracle is allowed throughout the game.
The ElGamal public-key encryption scheme

ElGamal’s encryption scheme is based on Diffie-Hellman. Let \( G = \langle g \rangle \) be a cyclic group of prime order \( q \).

Plaintexts and ciphertexts in the scheme are elements of \( G \).

The scheme \( \mathcal{E}-\text{ElGamal} = (G-\text{ElGamal}, E-\text{ElGamal}, D-\text{ElGamal}) \) is defined by:

\[
\begin{align*}
G-\text{ElGamal}: & \quad \alpha \xleftarrow{\$} \{0, 1, \ldots, q - 1\}; \\
& \quad \text{return} \ (pk = g^\alpha, sk = \alpha);
\end{align*}
\]

\[
\begin{align*}
E-\text{ElGamal}_{pk}(m): & \quad \beta \xleftarrow{\$} \{0, 1, \ldots, q - 1\}; \\
& \quad \text{return} \ (g^\beta, m \cdot pk^\beta);
\end{align*}
\]

\[
\begin{align*}
D-\text{ElGamal}_{sk}(y): & \quad (B, C) \leftarrow y; \\
& \quad m \leftarrow B^{-sk} C; \\
& \quad \text{return} \ m;
\end{align*}
\]

\[
\text{This scheme is secure in the IND-CPA sense if the Decisional Diffie-Hellman problem is hard in } G.
\]
Security proof for ElGamal

Suppose $\mathcal{A}$ is an adversary attacking the ElGamal scheme in the IND-CCA sense.

We construct from it an algorithm $\mathcal{D}$ which solves the DDH problem (i.e., given a triple $A = g^\alpha, B = g^\beta, C$, decides whether $C = g^{\alpha \beta}$):

Algorithm $\mathcal{D}(A, B, C)$:
1. $(m_0, m_1, s) \leftarrow \mathcal{A}(\text{find}, A)$;
2. $b \leftarrow \{0, 1\}$;
3. $y \leftarrow (B, m_b \cdot C)$;
4. $b' \leftarrow \mathcal{A}(\text{guess}, y, s)$;
5. if $b = b'$ then return 1; else return 0;

Security proof for ElGamal (cont.)

Let $\alpha$ and $\beta$ be the discrete logs of $A$ and $B$.

- If $C = g^{\alpha \beta}$, then $\mathcal{D}$’s success probability is equal to $\mathcal{A}$’s probability of guessing the hidden bit correctly, which is

$$\text{Adv}^{\text{ind-CCA}}_{\mathcal{E}-\text{ElGamal}}^G(\mathcal{A}) + \frac{1}{2}.$$

- If $C$ is random, then $m_b C$ is uniformly distributed in $G$, and independent of $b$, so $\mathcal{A}$ answers correctly with probability exactly $\frac{1}{2}$.

Hence, $\text{Adv}^{\text{ddh}}_G(D) = \text{Adv}^{\text{ind-CCA}}_{\mathcal{E}-\text{ElGamal}}(\mathcal{A})/2$, and

$$\text{InSec}^{\text{ind-CCA}}_{\mathcal{E}-\text{ElGamal}}^G(t) \leq 2 \cdot \text{InSec}^{\text{ddh}}_G(t).$$

\[\square\]
Notes about ElGamal

- We needed the Decisional Diffie-Hellman assumption to prove the security. This is a strong assumption. Still, a proof based on DDH is a lot better than nothing.

- We really do need the Decisional Diffie-Hellman assumption.
  
  An adversary with a DDH algorithm can submit \( m_0 \in_R G \) and \( m_1 = 1 \); it receives a ciphertext \((B, C)\), and returns 1 if \((A, B, C)\) looks like a Diffie-Hellman triple, or 0 if it looks random.

- The plaintexts must be elements of the cyclic group \( G \).
  
  For example, if \( G \) is a subgroup of \( \mathbb{F}_p^* \), it’s not safe to allow elements outside the subgroup as plaintexts: an adversary can compare orders of ciphertext elements to break the semantic security of the scheme.

- ElGamal is malleable. We can decrypt a challenge ciphertext \( y = (g^\beta, A^\beta x) \) by choosing a random \( \gamma \) and requesting a decryption of \( y' = (g^{\beta \gamma}, A^{\beta \gamma} x^\gamma) \).

Random Oracle Model

- idealized model introduced by Bellare and Rogaway in 1993

- considers cryptographic constructions that make use of a function \( H \)
  
  - can be accessed in a black-box way
  - answers consistently for values \( x \) already queried
  - for new values \( x \), choose uniformly at random in the range as answer

- Do they exist?
  
  \( \sim \text{ NO!} \) But let us assume cryptographic hash functions behave “approximately” like ROs
Random Oracle Model

Why ROM?
- allows efficient constructions of cryptographic primitives with somewhat “provable security” guarantees
- Efficient signature and encryption schemes (Schnorr signatures, . . . )

How are ROs used in security proofs?
- Sample a random $H$ at the beginning of an experiment
- Output of ROM fully hidden unless queried, i.e., $H(m, r)$ for $r$ a large random string
- Typically we assume that the reduction can “program” the random oracle i.e., can choose the answers to the oracle calls

Criticism of the ROM
- only a “heuristic” argument for security instead of a real proof
- There are schemes that can be shown secure in the ROM, but insecure when ROM is replaced with any real hash function

The Hash ElGamal public-key encryption scheme

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of order $q$.
Let $\mathcal{H} : \mathbb{G} \rightarrow \{0, 1\}^\ell$ be an hash function.

Plaintexts are elements of $\{0, 1\}^\ell$.

\begin{align*}
\text{G-H-ElGamal:} & \quad \text{E-H-ElGamal}_{pk}(m): \\
\alpha \xleftarrow{} \{0, 1, \ldots, q-1\}; & \quad \beta \xleftarrow{} \{0, 1, \ldots, q-1\}; \\
\text{return } (pk = g^\alpha, sk = \alpha); & \quad \text{return } (g^\beta, m \oplus \mathcal{H}(pk^\beta)); \\
\text{D-H-ElGamal}_{sk}(y): & \quad (B, C) \leftarrow y; \\
& \quad m \leftarrow \mathcal{H}(B^{sk}) \oplus C; \\
& \quad \text{return } x;
\end{align*}

This scheme is secure (in the Random Oracle Model) in the IND-CPA sense if the Computational Diffie-Hellman problem is hard in $\mathbb{G}$. 
The Hash ElGamal public-key encryption scheme

Let $G = \langle g \rangle$ be a cyclic group of order $q$.
Let $H : G \rightarrow \{0,1\}^\ell$ be an hash function.
Let $\mathcal{G} : G \times \{0,1\}^\ell \rightarrow \{0,1\}^k$ be an hash function.

Plaintexts are elements of $\{0,1\}^\ell$.

\[
\begin{align*}
G-H+\text{-ElGamal}: & \\
\alpha & \leftarrow \{0,1,\ldots,q-1\}; \\
\text{return} & \ (a = g^{\alpha}, \alpha);
\end{align*}
\]
\[
\begin{align*}
E-H+\text{-ElGamal}_{pk}(x): & \\
\beta & \leftarrow \{0,1,\ldots,q-1\}; \\
\text{return} & \ (g^{\beta}, x \oplus H(pk^{\beta}), G(x, pk^{\beta}));
\end{align*}
\]
\[
\begin{align*}
D-H+\text{-ElGamal}_{sk}(y): & \\
(B, c, d) & \leftarrow y; \\
x & \leftarrow H(B^{sk}) \oplus c; \\
\text{return} & \ x \text{ if } d = G(x, B^{sk}); \\
\text{return} & \ \bot \text{ otherwise.}
\end{align*}
\]

This scheme is IND-CCA2 (in the Random Oracle Model) if the (strong) Computational DH problem is hard in $G$.

Digital Signatures

- A very important public key primitive is the digital signature.

- The idea is
  - Message + Alice’s Private Key = Signature
  - Message + Signature + Alice’s Public Key = YES/NO

- Alice can sign a message using her private key.

- Anyone can verify Alice’s signature, since everyone can obtain her public key.

- the verifier is convinced that only Alice could have produced the signature
  - only Alice knows her private key!
Digital signature schemes

**Digital signatures**: Alice owns two “keys”
- a *public* key known by everybody (including Bob)
- a *secret* key known by Alice only

![Diagram showing Alice and Bob with keys and signatures]

**Digital Signatures: Services**

- The verification algorithm is used to determine whether or not the signature is properly constructed.

- the verifier has guarantee of
  - message *integrity* and
  - message *origin*.

- also provide *non-repudiation* - not provided by MACs.

*Most important cryptographic primitive!*
Security Notions

Depending on the context in which a given cryptosystem is used, one may formally define a security notion for this system,

- by telling what goal an adversary would attempt to reach,
- and what means or information are made available to her (the attack model).

A security notion (or level) is entirely defined by pairing an adversarial goal with an adversarial model.

Examples: \(\text{UB-KMA, UUF-KOA, EUF-SOCMA, EUF-CMA}\).

Signature Schemes

An digital signature scheme is a triple of algorithms \((G, S, V)\) where

- \(K\) is a probabilistic key generation algorithm which returns random pairs of secret and verification keys \((sk, vk)\) depending on the security parameter \(\kappa\),

- \(S\) is a (probabilistic) signature algorithm which takes on input a signing key \(sk\) and a message \(m \in M\), runs on a random tape \(u \in U\) and returns \(s \in S\),

- \(V\) is a deterministic verification algorithm which takes on input a verification key \(vk\), a message \(m\) and \(s \in S\) and outputs a bit in \(\{0, 1\}\).

If \(V_{\|} (\$, f) = \infty\), then \(s\) is a signature on \(m\) for \(vk\).

If \((sk, vk) \leftarrow K\), then \(V_{vk} (m, S_{sk} (m, u)) = 1\) for all \((m, u) \in M \times U\).
Security Goals

[Unbreakability] the attacker recovers the secret key $sk$ from the public key $vk$ (or an equivalent key if any). This goal is denoted $UB$. Implicitly appeared with public-key cryptography.

[Universal Unforgeability] the attacker, without necessarily having recovered $sk$, can produce a valid signature of any message in the message space. Noted $UUF$.

[Existential Unforgeability] the attacker creates a message and a valid signature of it (likely not of his choosing). Denoted $EUF$.

Adversarial Models

- **Key-Only Attacks** (KOA), unavoidable scenario.

- **Known Message Attacks** (KMA) where an adversary has access to signatures for a set of known messages.

- **Chosen-Message Attacks** (CMA) the adversary is allowed to use the signer as an oracle (full access), and may request the signature of any message of his choice
Chosen-Message Security

Goldwasser, Micali, Rivest (1988)
A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks.

Formally, a signature scheme is said to be \((q, \tau, \varepsilon)\)-secure if for any adversary \(A\) with running time upper-bounded by \(\tau\),

\[
\text{Succ}^{\text{EUF-CMA}}(A) = \Pr \left[ \begin{array}{l}
(\sk, \vk) \leftarrow \mathcal{K}(1^k), \\
(\m^*, \s^*) \leftarrow A^S(\sk, \cdot)(\pk), \\
\mathcal{V}(\vk, \m^*, \s^*) = 1
\end{array} \right] < \varepsilon,
\]

where the probability is taken over all random choices.

The notation \(A^S(\sk, \cdot)\) means that the adversary has access to a signing oracle throughout the game, but at most \(q\) times.

The message \(\m^*\) output by \(A\) was never requested to the signing oracle.

EUF-CMA: Playing the Game

\[G(1^k)\]

Key Generator

\(sk\)

\(pk\)

\(A\)

Signing Oracle

\(S(\sk, \cdot)\)

\(m^*, s^*\)

Verification

\[V(pk, \cdot)\]

1?
Lamport signatures

L. Lamport
Constructing digital signatures from a one-way function

- a Lamport signature or Lamport one-time signature scheme is a method for constructing efficient digital signatures.
- Lamport signatures can be built from any cryptographically secure one-way function; usually a cryptographic hash function is used.
- Unfortunately each Lamport key can only be used to sign a single message.
- However, we will see how a single key could be used for many messages.

How to sign one bit just once?

\( \mathcal{M} = \{0, 1\} \)

- **Key generation:**
  - Consider \( f : X \rightarrow Y \) a one-way function.
  - e.g.
    \[
    f : \mathbb{Z}_q \rightarrow \mathbb{G} \\
    x \mapsto f(x) = g^x r
    \]
  - Select two random elements \( x_0, x_1 \in X \).
  - Compute their images \( y_i = f(x_i) \) for \( i \in \{0, 1\} \).
  - **Verification key** \( \text{vk} = (y_0, y_1) \) which can be published.
  - **Signing key** \( \text{sk} = (x_0, x_1) \) which needs to be kept secret

- **Signature:** if Alice wants to sign a bit \( b \), she does the following:
  - Use her signing key \( (x_0, x_1) \) to send the signature \( s = x_b \) to Bob.
- **Verification:** to check the validity of \( s \) on \( b \), Bob does the following:
  - Obtain Alice’s authentic verification key \( (y_0, y_1) \).
  - Check whether \( f(s) = y_b \).
How to sign \( k \) bits \textbf{just once}?

\[
\mathcal{M} = \{0,1\}^k
\]

- **Key generation:**
  - Generate \( f : X \rightarrow \) a \textbf{one-way function}.
  - Select \( 2k \) random elements \( x_{0,1}, x_{1,1}, \ldots, x_{0,k}, x_{1,k} \in X \).
  - Compute their images \( y_{i,j} = f(x_{i,j}) \) for \( i \in \{0,1\} \) and \( j \in [1,k] \).

  **Verification key** \( vk = (y_{0,1}, y_{1,1}, \ldots, y_{0,k}, y_{1,k}) \) which can be published.

  **Signing key** \( sk = (x_{0,1}, x_{1,1}, \ldots, x_{0,k}, x_{1,k}) \) which needs to be kept secret

- **Signature:** if Alice wants to sign \( m = m_1 \ldots m_k \), she does the following:
  - Use her signing key \( (x_{0,1}, x_{1,1}, \ldots, x_{0,k}, x_{1,k}) \) to send the signature \( s = (x_{m_1,1}, x_{m_1,2}, \ldots, x_{m_k,k}) \) to Bob.

- **Verification:** to check the validity of \( s = (s_1, \ldots, s_k) \) on \( m \), Bob does the following:
  - Obtain Alice’s authentic verification key \( (y_{0,1}, y_{1,1}, \ldots, y_{0,k}, y_{1,k}) \).
  - Check whether \( f(s_i) = y_{m_i,i} \) for all \( i \in [1, k] \).

**Theorem**

\textit{If} \( f \) \textit{is} \((\tau, \varepsilon)\)-\textit{one way then Lamport’s signature scheme (for} \( k \)-\textit{bit messages) is} \((1, \tau', 2k \cdot \varepsilon)\)-\textit{EUF-CMA secure, with} \( \tau' = \tau + (2k - 1)T_{\text{Eval}} \).

- In other words: If there is an Adversary \( \mathcal{A} \) that chooses
  - a message \( m \in \{0,1\}^k \) for Alice to legitimately authenticate
  - forges a message \( m' \neq m \) with probability at least \( \varepsilon \)

Then there is an Adversary \( \mathcal{B} \) that can break the one-wayness of the function \( f \) with probability at least \( \varepsilon/2k \) operates in time roughly the same as \( \mathcal{A} \)
How to sign \( k \) bits just once?

**Proof.** \( B \) gets as input the description of \( f \) and \( y^* \in Y \).

- \( B \) picks as input an index \((i^*, j^*) \in \{0, 1\} \times \{1, k\}\)
- \( B \) selects \( 2k - 1 \) random elements \( x_{0,1}, \ldots, x_{i^*, j^*}, \ldots, x_{1,k} \in X \).
- \( B \) computes their images \( y_{i,j} = f(x_{i,j}) = \text{Eval}(x_{i,j}) \) for \((i, j) \in \{0, 1\} \times \{1, k\} \setminus \{(i^*, j^*)\}\).
- \( B \) sets \( y_{i^*, j^*} = y \)
- \( B \) executes \( A \) on the public key \((y_{0,1}, y_{1,1}, \ldots, y_{0,k}, y_{1,k})\)
- At some point \( A \) query one message \( m = m_1 \ldots m_k \) to the signature oracle
  - If \( m_{j^*} = i^* \) then \( B \) aborts the simulation (probability \( 1/2 \)),
  - otherwise \( B \) outputs a valid signature on \( m \) thanks to its knowledge of \( x_{0,1}, \ldots, x_{i^*, j^*}, \ldots, x_{1,k} \).
- Eventually, \( A \) outputs a signature \( s' \) on a message \( m' \neq m \) and \( B \) outputs \( s'_{j^*} \).
  The message \( m' \) differs from \( m \) in at least one position. If it is the \( j^* \)-th position (probability \( 1/k \)) and if the signature is valid (probability \( \varepsilon \)) we have \( f(s'_{j^*}) = y_{i^*, j^*} = y \).

\[
\square
\]
Lamport’s signatures: variants

- **Short private key.** Instead of creating and storing all the random numbers of the private key a **single key** of sufficient size can be stored.

  The single key can then be used as the seed for a **cryptographically secure pseudorandom number generator** to create all the random numbers in the private key when needed.

- **Short public key** A Lamport signature can be combined with a **hash list**, making it possible to only publish a single hash instead of all the hashes in the public key.

- **Hashing the message.**
  - Unlike some other signature schemes the Lamport signature scheme **does not** require that the message $m$ is hashed before it is signed.
  - A system for signing long messages can use a collision resistant hash function $h$ and sign $h(m)$ instead of $m$.

### Lamport’s signatures: several messages

![Diagram of Lamport's signatures for several messages]
Groth’s one-time signatures

Groth (2006)
Simulation-sound NIZK proofs for a practical language and constant size group signatures.

Key generation: generate $vk = (X = g^x, Y = g^y, Z = g^z)$ where $x, y \overset{\$}{\leftarrow} \mathbb{Z}_p^*$

Sign: to sign $m \in \mathbb{Z}_p^*$, select $r \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, compute
$s = (1 - mx - yr)/z \in \mathbb{Z}_p^*$, and output $\sigma = (r, s)$.

Verify: given $\sigma \in (\mathbb{Z}_p^*)^2$, check
$X^m Y^r Z^s = g$.

Theorem
If the discrete logarithm assumption holds in $\mathbb{G}$ then Groth’s signature scheme is one-time EUF-CMA secure.

Proof idea: given a DL instance $(g, h) \in \mathbb{G}$, one sets $X = g^{a_1} h^{b_1}$, $Y = g^{a_2} h^{b_2}$, $Z = g^{a_3}$ where $a, b, c \overset{\$}{\leftarrow} \mathbb{Z}_p^*$. On signature query on $m$, one compute
$r = -mb_1/b_2 \mod p$ and $s = (1 - ma_1 - r_2)/a_3 \mod p$.

Thanks to the adversary’s forgery, one can retrieve the discrete logarithm of $h$ in base $g$ by solving a linear system.
Graph isomorphism

- In **graph theory**, an **isomorphism** of graphs $G$ and $H$ is a bijection between the vertex sets of $G$ and $H$

  $$f : V(G) \rightarrow V(H)$$

  such that any two vertices $u$ and $v$ of $G$ are adjacent in $G$ if and only if $f(u)$ and $f(v)$ are adjacent in $H$.

- If an isomorphism exists between two graphs, then the graphs are called **isomorphic**.

- The computational problem of determining whether two finite graphs are isomorphic is referred to as the **graph isomorphism problem**.

- The graph isomorphism problem is a curiosity in computational complexity theory: not known to be in $\mathcal{P}$ nor $\mathcal{NP}$-complete.
Graph isomorphism

Graph isomorphism
Zero-knowledge interactive proof

- A zero-knowledge proof or zero-knowledge protocol is an interactive method for one party to prove to another that a (usually mathematical) statement is true, without revealing anything other than the veracity of the statement.

- A zero-knowledge proof must satisfy three properties:
  - **Completeness**: if the statement is true, the honest verifier (that is, one following the protocol properly) will be convinced of this fact by an honest prover.
  - **Soundness**: if the statement is false, no cheating prover can convince the honest verifier that it is true, except with some small probability.
  - **Zero-knowledge**: if the statement is true, no cheating verifier learns anything other than this fact.

- The first two of these are properties of more general interactive proof systems. The third is what makes the proof zero-knowledge.

---

Zero-knowledge interactive proof for Graph Isomorphism

**Input**: Two graphs $G_0$ and $G_1$ each having vertex set $\{1, \ldots, n\}$. Alice knows $\sigma \in \mathcal{S}_n$, an isomorphism from $G_0$ to $G_1$.

**Repeat the following** $n$ times
- Alice chooses a random permutation $\pi \in \mathcal{S}_n$.
- She computes $H$ to be the image of $G_0$ under $\pi$ and sends $H$ to Bob.
- Bob chooses randomly $b \in \{0, 1\}$ and sends it to Alice.
- Alice sends $\rho = \pi \circ \sigma^b$ to Bob.
- Bob checks if $H$ is the image of $G_b$ under $\rho$. 
Schnorr’s ID Protocol (1989)

Let $G = \langle g \rangle$ be a group of prime order $q$.

Prover $P$ proves to verifier $V$ that he knows the discrete log $x$ of a public group element $y = g^x$. It is a 3-move protocol.

\[
\begin{align*}
  & x \leftarrow \mathbb{Z}_q \\
  & y = g^x \\
  & k \leftarrow \mathbb{Z}_q \\
  & r = g^k \\
  & s = k + cx \mod q
\end{align*}
\]

\[
\begin{align*}
  P & \quad V \\
  r & \quad c \\
  s & \quad g^s \cdot y^{-c} = r
\end{align*}
\]

Scenario

- $P$ sends $r = g^k$ where $k \leftarrow \mathbb{Z}_q$
- $V$ sends $c \leftarrow \mathbb{Z}_q$
- $P$ sends $s = k + cx \mod q$
- $V$ checks whether $g^s \cdot y^{-c} = r$

The Fiat-Shamir heuristic

Fiat, Shamir (1986)
How to Prove Yourself: Practical Solutions to Identification and Signature Problems.

In such a 3-pass identification scheme, the messages are called **commitment**, **challenge** and **response**. The challenge is randomly chosen by $V$.

**Fiat-Shamir Transform:** replace the challenge by a hash value taken on scheme parameters and $t$, thereby removing $V$. This transforms the protocol by making it non-interactive.

The intuition is that any "sufficiently random" hash function should preserve the security of the protocol.

(Many applications $\leadsto$ see Damien’s lectures / Luca’s lectures)
Schnorr Signatures (via the Fiat-Shamir Transform)

Introduce a hash function $H : \{0, 1\}^* \mapsto \mathbb{Z}_q$

Schnorr’s signature scheme $\mathcal{g}ma_H$ is a tuple of probabilistic algorithms $\mathcal{g}ma_H = (\text{Gen}, \text{Sign}, \text{Ver})$ defined as follows.

\[
x \xleftarrow{\$} \mathbb{Z}_q \\
y = g^x \\
\text{Gen}
\]

\[
k \xleftarrow{\$} \mathbb{Z}_q \\
r = g^k \\
\text{Sign}
\]

\[
s = k + cx \mod q \\
\sigma = (s, c)
\]

\[
\begin{array}{c}
m \\
H \\
\sigma = (s, c) \\
\end{array}
\]

\[
H(m, g^s \cdot y^{-c}) = c \\
\text{Ver}
\]

Security of Schnorr Signatures - Key Only Attacks

**Theorem**

*If there exist a $(0, \tau, \varepsilon)$-EUF-CMA adversary in the ROM (with $q_H$ queries to the RO) against Schnorr’s signature scheme (in $\mathbb{G}$), then the discrete logarithm in $\mathbb{G}$ can be solved in expected time $O(\tau \cdot q_H / \varepsilon)$.***

**Proof Intuition**

- run the adversary $A$ several times in related executions
- the process “forks” at a certain point (modification of the RO)
- hope for two executions of $A$ with forgery on the same message queried to the RO (but with different hash values)
  ~~~ extract the discrete logarithm
Security of Schnorr Signatures - Chosen Message Attacks

**Theorem**

If there exist a \((q_S, \tau, \varepsilon)\)-EUF-CMA adversary in the ROM (with \(q_H\) queries to the RO) against Schnorr’s signature scheme (in \(G\)), then the discrete logarithm in \(G\) can be solved in expected time \(O(\tau \cdot q_H/\varepsilon)\).

The previous result can be adapted readily for an EF-CMA adversary.

In order to answer signing queries, one simply uses the simulator of the zero-knowledge proof \((r, h, s)\), and we set \(H(m, r) := h\). The random oracle programming may fail, but with negligible probability.

**References**